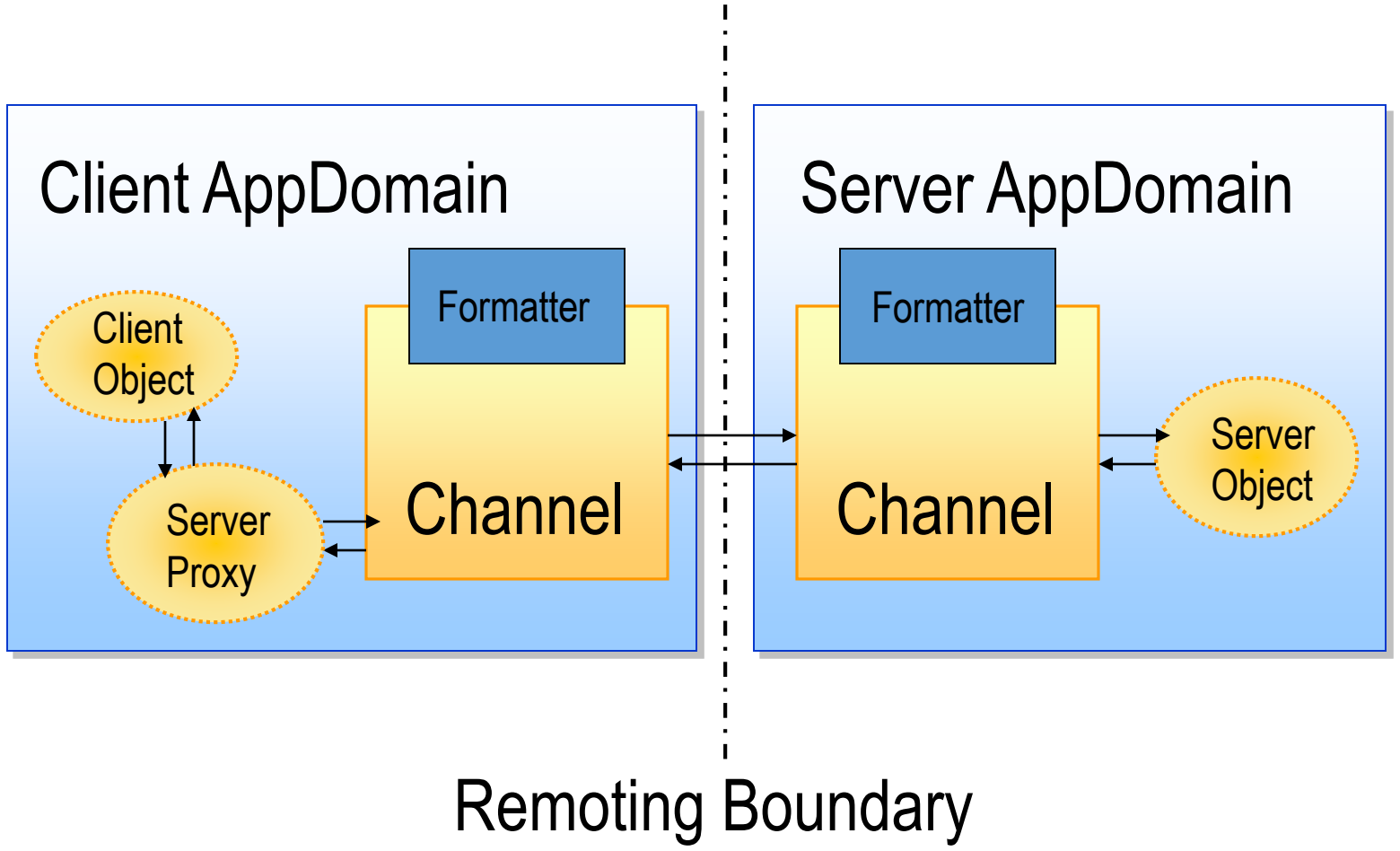
.NET Remoting
XML Web Service

.NET Remoting

- Remoting Overview
- Channels and Formatters
- Activation and Proxies
- Lease-Based Lifetime
- Object Marshaling
- Server Side
- Client Side
- Client Compilation Techniques

Remoting Overview



Channels and Formatters

- Channels Transport Messages To and From Remote Objects
- Client Selects a Channel That Is Registered on the Server
  - Before calling a remote object, the client registers the channel Channels are registered on a per application domain basis
  - One computer cannot have multiple channels listening to same port
- .NET Provides Implementation of HTTP and TCP Channels
  - HTTP Channel default: SOAP protocol to transport XML messages
  - TCP Channel default:TCP protocol to transport binary messages

    Faster than HTTP SOAP Web Services but not as open
- Example: Programmatic Registration of TCP Channel on Port 8085

```
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
        TcpChannel chan = new TcpChannel(8085);
        ChannelServices.RegisterChannel(chan);
```

Activation and Proxies

- Before Using a Remote Object, the Client Must Activate It
  - By calling **new**, **Activator.CreateInstance**, or **Activator.GetObject**
- Activation Returns Proxy Used by Client to Access Remote Object
  - Proxy represents remote object in client's AppDomain
  - Proxy forwards client's calls, and returns results and exceptions
- Server-Side Activation – Automatic Instantiation by Server
  - Single call object handles only one request (stateless)
  - Singleton object services multiple clients and requests (stateful)
- Client-Side Activation – Instantiation by Explicit Client Call
  - State maintained between method calls for specific client instance

Remoting - Server Side

- Register the Channel
- Register Remote Objects by Using:
  - The **RegisterWellKnownServiceType** call

```
RemotingConfiguration.RegisterWellKnownServiceType(
      typeof(HelloServer),
      "SayHello",
      WellKnownObjectMode.SingleCall);
```

  - Or a configuration file

```
RemotingConfiguration.Configure("MyHello.exe.config");
```

Remoting - Client Side

- Register the Channel

```
ChannelServices.RegisterChannel(new TcpChannel());
```

- Activate Remote Object by Using:
  - **Activator.GetObject**

```
HelloServer obj = (HelloServer)Activator.GetObject(
      typeof(RemotingSamples.HelloServer),
      "tcp://localhost:8085/SayHello");
```

  - **new** and a configuration file

```
RemotingConfiguration.Configure("MyHello.exe.config");
HelloServer obj = new HelloServer();
```

Client Compilation Techniques

- When the Client Is Compiled, the Compiler Needs Server Class Data
- Class Information Can Be Provided by:
  - A reference to the assembly where the class is stored
  - Splitting the remote object into an implementation class and an interface type
  - Using Wsdl.exe to extract the required metadata directly from the endpoint

Remoting Configuration Files

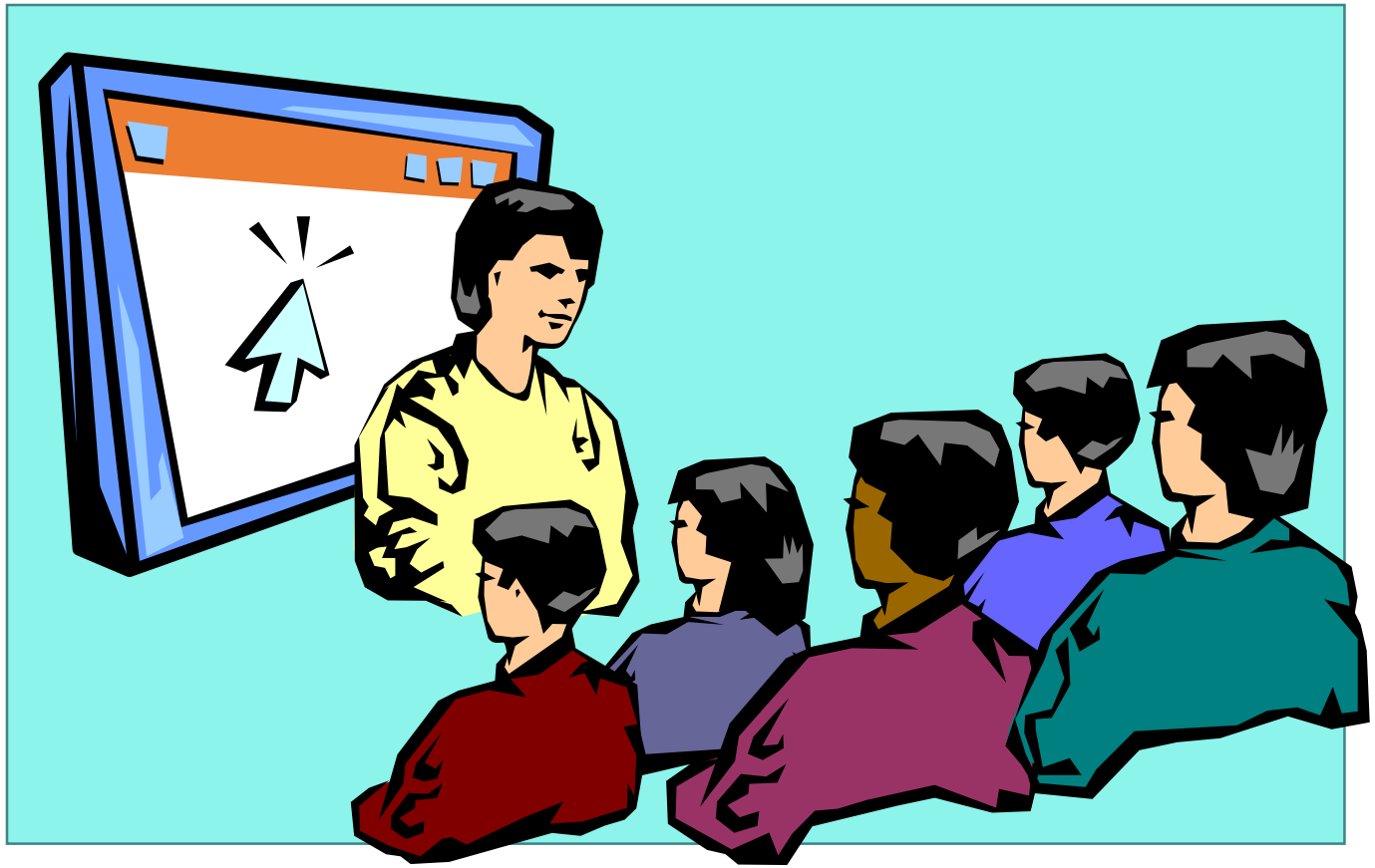
- Configure Method on RemotingConfiguration

```
RemotingConfiguration.Configure(foo.exe.config);
```

- An Application Configuration File is an XML Document
  - Configuration files are case-sensitive
  - Can be specified on a machine or on an application level

    Application level configuration takes priority over machine level configuration
  - For more information see “Remoting Configuration File Format” in the .NET Framework SDK documentation

Demonstration: Remoting



Object Marshaling

- Objects Instantiated Remotely Are Returned by Reference and Accessed by the Client Through a Proxy
- Remote Call Parameters, Return Values, and Fields Can Be:
  - Marshal-by-value objects – A copy of the object is passed from one AppDomain to another
    - - Value types and classes that are serializable
  - Marshal-by-reference objects – A reference to the object is passed from one AppDomain to another
    - - Classes that derive from the **System.MarshalByRefObject** class
  - Not-marshaled objects – Objects suitable for local use only
    - - Any class that is not Marshal-By-Value or Marshal-By-Reference

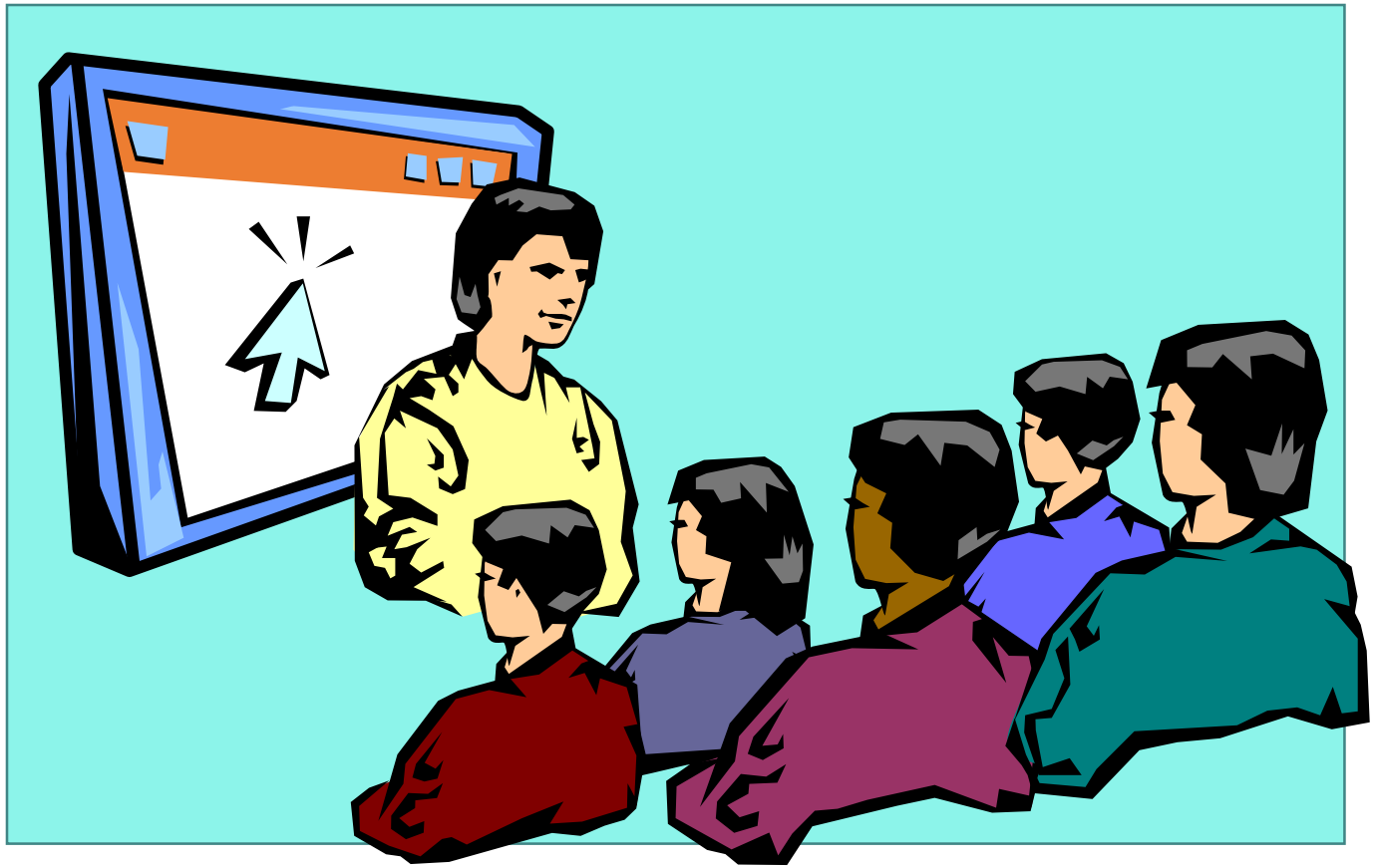
Using Marshal by Value

- A remoting method can pass data by value
  - All data for the class passes through the chanell
  - A serializer is associated with the channel
- There are two build-in formatters
  - BinaryFormatter
  - SoapFormatter
- When passed by value, methods are never remote
- A user-defined class must have Serializable attribute

Using Marshal by Reference

- The object is created on the server and stays there
- Client receives a reference to the object
- Methods are executed remotely
- The remoted type must inhered from MarshalByRefObject

Demonstration: Object Marshaling



Using Events in Remoting Applications

- Event Listener
  - Listener is marshaled by reference
  - Delegate from event listener passed to server’s event handler
  - Server raises event and calls client
- Channel Configuration
- Security Recommendations
  - Strong names
  - Authentication
  - Encryption
- Server Failure Notification

Demonstration: Using Events in Remoting Applications